# STRUCTURE PRESERVING TRANSFORMATIONS ON NON-LEFT-RECURSIVE GRAMMARS

(preliminary version)

Anton Nijholt
Vrije Universiteit
Department of Mathematics
P.O.-Box 7161, Amsterdam
The Netherlands

## 1. INTRODUCTION AND PRELIMINARIES

If a context-free grammar is transformed to another context-free grammar in most of the cases it is quite obvious to demand *weak equivalence* for these two grammars. Transformations on context-free grammars can be defined for several reasons. Dependent on these reasons one may be interested in stronger relations of grammatical similarity.

Instead of arbitrary context-free grammars one can consider context-free grammars which conform to some requirements on, for example, the form of the productions. It is natural to ask whether each context-free language has a context-free grammar in this form and, if possible, how to transform a context-free grammar to a context-free grammar of this special form.

One of the reasons to consider normal forms may be the mathematical interest in how to generate a class of languages with a so simple possible grammatical description. Moreover, normal forms can simplify descriptions and proofs. Some normal form descriptions of the context-free grammars or their subclasses can be particularly amenable for parsing and this can be a strong motivation to transform grammars. Transformations can be applied to obtain grammars for which smaller sized parsers or faster parsing methods can be constructed. For such transformations stronger relations than weak equivalence are desirable.

A slightly stronger relation is obtained if we demand that the language preserving transformation is such that each sentence has the same number of parse trees in each grammar, that is, the transformation is also *ambiguity preserving*.

Another relation which has been defined is *structural equivalence*, in which case it is demanded that the parse trees of the one grammar are the same, except for a relabeling of the internal nodes, as the trees of the other grammar.

Our interest is in the semantic equivalence of context-free grammars which are syntactically related. It is assumed that semantic rules are associated with each production of a grammar and, quite obvious, it follows that we will be interested in the correspondence of the derivations of related grammars. Such a correspondence should be formalized.

Some rather independent developments can be distinguished.

a. In the older literature one can find ideas and examples which come close to later formal concepts, for example Griffiths and Petrick [15], Kurki-Suonio [27], Kuno [26] and Foster [8]. Transformations have been defined in practically oriented situations of compiler construction. In such cases no general definitions of the syntactic relation between the grammars are presented.

b. *Grammar functors* (X-functors) were introduced by Hotz [20,21] as special functors on categories associated with (general) phrase structure grammars. These syntax categories originate from work on switching circuits. The main concern has been to find an algebraic framework for describing general properties of phrase structure grammars. Only recently functors have been considered from a more "practical" point of view. See for example Bertsch [3], Benson [2], Walter, Keklikoglou and Kern [42] and Hotz [22].

c. *Grammar covers*, in the sense that we will use them here, were introduced about 1969 by Gray and Harrison [11]. A practical reason to consider covers concerns compiler construction. In such a case we consider a parse as the argument of a semantic mapping. In case a context-free grammar G' covers a context-free grammar G we can use the original semantic mapping, corresponding to G, and do the parsing according to G'.

d. In the case of attribute grammars (see Knuth [24]) attributes are associated with the nodes of a parse tree. These attributes (which contain semantic information) are obtained from attributes associated with the symbols which appear in the productions and from attribute evaluation rules. If an attribute grammar is transformed to, for example, some normal form attribute grammar, we have not only the question of language equivalence but also the question of semantic equivalence. Such an equivalence is explored in Bochman [4,5].

We will be concerned with grammar covers. The first part of this paper presents a general framework for covers. The second part introduces a transformation from non-left-recursive grammars to grammars in Greibach normal form. An investigation of the structure preserving properties of this transformation, which serves also as an illustration of our framework for covers, is presented.

## Preliminaries

We shortly review some definitions and concepts of formal language theory. It is assumed that the reader is familiar with the basic results concerning context-free grammars and with parsing, otherwise see Aho and Ullman [1].

Let V be an *alphabet* and let $\alpha \in V^*$, then $|\alpha|$ denotes the *length* of string $\alpha$. The *empty* string is denoted by $\varepsilon$. If $|\alpha| \geq k$ then $\alpha : k$ denotes the *suffix* of $\alpha$ with

length k, otherwise $\alpha : k = \alpha$. For *prefixes* the notation $k : \alpha$ is used. The number of elements in any set V is denoted by $|V|$; the empty set by $\emptyset$. $\mathbb{N}$ stands for the set of positive integers.

Consider two alphabets V en W. A *homomorphism* $h : V^* \to W^*$ is obtained by defining $h : V \to W^*$, $h(\varepsilon) = \varepsilon$ and $h(\alpha\beta) = h(\alpha).h(\beta)$ for all $\alpha, \beta \in V^*$. Let $\Sigma$ and $\Delta$ be disjoint alphabets. Homomorphism $h_\Sigma : (\Sigma \cup \Delta)^* \to \Delta^*$ is defined by $h_\Sigma(X) = X$ if $X \in \Delta$, and $h_\Sigma(X) = \varepsilon$ if $X \in \Sigma$. Homomorphism $h_\Sigma$ is called the *$\Sigma$-erasing homomorphism*.

A *context-free grammar* (CFG) will be denoted by the four-tuple $G = (N,\Sigma,P,S)$ where $N \cap \Sigma = \emptyset$, N is the set of *nonterminals*, $\Sigma$ is the set of *terminals*, $N \cup \Sigma$ is denoted by V, $S \in N$ is the *start symbol* and P is the set of *productions*, $P \subseteq N \times V^*$. Elements of N will generally be denoted by the Roman capitals A,B,C,...,S,...; elements of $\Sigma$ by the smalls a,b,c,... from the first part of the Roman alphabet; X, Y and Z will usually stand for elements of V, elements of $\Sigma^*$ will be denoted by u,v,w,x,y and z and Greek smalls $\alpha,\beta,\gamma,\ldots$ will usually stand for elements of $V^*$. It will be convenient to provide the productions in P with a label. In general these labels will be in a set $\Delta_G$ (or $\Delta$ if G is understood) and we always take $\Delta_G = \{i \mid 1 \leq i \leq |P|\}$; we often identify P and $\Delta_G$. We write $i.\ A \to \alpha$ if production $A \to \alpha$ has label (or number) i.

We have the usual notations $\Rightarrow$, $\overset{+}{\Rightarrow}$ and $\overset{*}{\Rightarrow}$ for *derivations* and we use subscripts L and R for leftmost and rightmost derivations, respectively. The notation $\overset{\pi}{\Rightarrow}$ will be used to denote that the derivation is done according to a specific sequence $\pi \in \Delta_G^*$ of production numbers. A *left parse* of a sentence $w \in L(G)$ (the language of G) is a sequence of productions used in a leftmost derivation from S to w. The reverse of a sequence used in a rightmost derivation is called a *right parse* of w.

The number of different leftmost derivations from S to w is called the *degree of ambiguity* of w (with respect to G), written $\langle w,G \rangle$.

The set of *parse trees* of G (with roots labeled with S and frontiers in $\Sigma^*$) is denoted by PTR(G). If $t \in PTR(G)$ then fr(t) denotes its frontier.

DEFINITION 1.1. A CFG $G = (N,\Sigma,P,S)$ is

a) *proper*, if G is $\varepsilon$-free, cycle-free and G has no useless symbols.

b) *left-recursive*, if there exist $A \in N$ and $\alpha \in V^*$ such that $A \overset{+}{\Rightarrow} A\alpha$.

c) in *Greibach normal form* (GNF), if $P \subseteq N \times \Sigma N^* \cup \{(S,\varepsilon)\}$.

It is rather natural to start with a CFG $G = (N,\Sigma,P,S)$ and generalize it to a *simple syntax directed translation scheme* (simple SDTS) $T = (N,\Sigma,\Delta,R,S)$, where $\Delta$(the output alphabet) contains the production numbers and R contains rules of the form $A \to \alpha,\alpha'$ where $A \to \alpha$ is in P and $\alpha'$ is a word over $(N \cup \Delta)^*$ which satisfies $h_\Delta(\alpha') = h_\Sigma(\alpha)$.

In such a case we say that T is defined on G. The *translation* defined by such a scheme T is denoted by $\tau(T)$.

DEFINITION 1.2. A simple SDTS is *semantically unambiguous* if there are no two distinct rules of the form $A \to \alpha,\beta$ and $A \to \alpha,\gamma$.

## 2. GRAMMAR COVERS

This section is devoted to building a general framework for grammar covers.
Let $G = (N,\Sigma,P,S)$ be a CFG with production numbers in $\Delta_G$. The following definition is also in Brosgol [6].

DEFINITION 2.1. A relation $f_G \subset \Sigma^* \times \Delta_G^*$ is said to be a *parse relation* for G provided that

(i) if $(w,\pi) \in f_G$ and $(w',\pi) \in f_G$ then $w = w'$, and

(ii) for each $w \in \Sigma^*$, $|\{\pi \mid (w,\pi) \in f_G\}| = |\{t \in PTR(G) \mid w = fr(t)\}|$.

If $f_G$ is a parse relation and $(w,\pi) \in f_G$ then $\pi$ is said to be an $f_G$-parse of w. Our following definitions will be based on parse relations. Index G of $f_G$ will be omitted whenever it is clear from context for which grammar $f_G$ is the parse relation.

DEFINITION 2.2. Let $G = (N,\Sigma,P,S)$ and $G' = (N',\Sigma',P',S')$ be CFGs. Let $f_{G'} \subseteq \Sigma'^* \times \Delta_{G'}^*$, and $h_G \subset \Sigma^* \times \Delta_G^*$ be parse relations. For a given $f_{G'}$ and $h_G$ a *parse homomorphism* $g : f_{G'} \to h_G$ is defined by two homomorphisms $\phi : \Sigma'^* \to \Sigma^*$ and $\psi : \Delta_{G'}^* \to \Delta_G^*$ such that $(w,\pi) \in f_{G'}$ implies $(\phi(w),\psi(\pi)) \in h_G$. If $\Sigma' = \Sigma$ and $\phi$ is the identity homomorphism then g is said to be externally fixed. We use the notation $g = \langle\phi,\psi\rangle$.

DEFINITION 2.3. A parse homomorphism $g : f_{G'} \to h_G$, with $g = \langle\phi,\psi\rangle$, is said to be a *cover homomorphism* if it is surjective, that is, for all $(w,\pi) \in h_G$ there exists $(w',\pi') \in f_{G'}$ such that $(w,\pi) = (\phi(w'),\psi(\pi'))$.

Notice that in the case of a cover homomorphism $\langle w,G'\rangle$ and $\langle\phi(w),G\rangle$ are incomparable. In case $\phi$ is the identity homomorphism then $L(G) = L(G')$ and $\langle w,G'\rangle \geq \langle\phi(w),G\rangle$. Notice that $\langle w,G'\rangle = |\{\pi \mid (w,\pi) \in f_{G'}\}|$.

DEFINITION 2.4. A parse homomorphism $g : f_{G'} \to h_G$ with $g = \langle\phi,\psi\rangle$, is said to be *properly injective* if its restrictions to $\Sigma'^*$ and $\Delta_{G'}^*$, are injective, that is,

(i) if $(w,\pi) \in f_{G'}$ and $(w,\pi') \in f_{G'}$, then $\psi(\pi) = \psi(\pi')$ implies $\pi = \pi'$, and

(ii) if $(u,\pi) \in f_G$, and $(v,\pi') \in f_G$, then $\phi(u) = \phi(v)$ implies $u = v$.

EXAMPLE 2.1

Let G' be defined (In our example grammars only the productions are listed.) by

| | | | | | |
|---|---|---|---|---|---|
| 0./1. | $S \to aA \mid cB$ | and G by | 0./1. | $S \to Ab \mid Bb$ |
| 2./3. | $A \to aA \mid b$ | | 2./3. | $A \to Aa \mid a$ |
| 4./5. | $B \to cB \mid d$ | | 4./5. | $B \to Ba \mid a$ |

Let $f_{G'} = \{(a^{n+1}b, 02^n3) \mid n \geq 0\} \cup \{(c^{n+1}d, 14^n5) \mid n \geq 0\}$

and $h_G = \{(a^{n+1}b, 32^n0) \mid n \geq 0\} \cup \{(a^{n+1}b, 54^n1) \mid n \geq 0\}$.

Homomorphism $g = \langle\phi,\psi\rangle$ is defined by

| | | | | |
|---|---|---|---|---|
| $\phi(a) = a$ | $\phi(c) = a$ | $\psi(0) = 3$ | $\psi(2) = 2$ | $\psi(4) = 4$ |
| $\phi(b) = b$ | $\phi(d) = b$ | $\psi(1) = 5$ | $\psi(3) = 0$ | $\psi(5) = 1$. |

It follows that g is a parse homomorphism which is surjective, hence g is a cover homomorphism. A parse homomorphism is said to be a *proper bijection* if it is both properly injective and surjective.

The results in the following table are immediate from the definitions given above.

| | | |
|---|---|---|
| surjection | - | $\phi(L(G')) = L(G)$ |
| proper injection | $\langle w,G'\rangle \leq \langle\phi(w),G\rangle$ | $\phi(L(G')) \subseteq L(G)$ |
| proper bijection | $\langle w,G'\rangle = \langle\phi(w),G\rangle$ | $\phi(L(G')) = L(G)$ |

Table I. Properties of parse homomorphisms.

In the following diagram the definition of a parse homomorphism is illustrated.

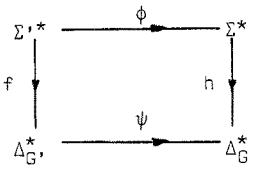


Figure 1. Diagram for the parse homomorphism.

Notice, if g is a cover homomorphism and $\phi$ is the identity homomorphism, then $L(G') = L(G)$ and $\langle w,G'\rangle \geq \langle w,G\rangle$.

DEFINITION 2.5. Let $G' = (N',\Sigma',P',S')$ and $G = (N,\Sigma,P,S)$ be two CFGs. Let $f \subseteq \Sigma'^* \times \Delta_{G'}^*$, and $h \times \Sigma^* \to \Delta_G^*$ be parse relations. G' *f-to-h covers* G if a cover homomorphism $g : f \to h$ can be defined.

Most of the time one will be satisfied with a cover homomorphism $g = \langle\phi,\psi\rangle$ such that $\phi : \Sigma'^* \to \Sigma^*$ is the identity homomorphism. In such cases there is only one homomorphism to consider, namely $\psi : \Delta_{G'}^* \to \Delta_G^*$ and we will simply speak of cover homomorphism $\psi$.
Examples of often used parses are left parses, right parses and left-corner parses (Rosenkrantz and Lewis [37]. These parses are examples of 'production directed' parses. In the following definition $h_\Sigma : (N \cup \Sigma \cup \Delta)^* \to (N \cup \Delta)^*$ is the $\Sigma$-erasing homomorphism.

DEFINITION 2.6. Let $G = (N,\Sigma,P,S)$ be a CFG. A parse relation $f \subseteq \Sigma^* \times \Delta^*$ is said to be a *production directed parse relation* for G if there exists a simple SDTS $T = (N,\Sigma,\Delta,R,S)$ where R is defined by : if $A \to \alpha$ is the ith production in P then R contains exactly one rule of the form $A \to \alpha$, $h_\Sigma(\alpha_1 i \alpha_2)$ with $\alpha_1\alpha_2 = \alpha$, R does not contain other rules, and $f = \tau(T)$.

It is a well-known trick to insert special symbols (standing for production numbers or, more generally, marking the place for semantical information) in the right-hand sides of productions to obtain special parses (for example, see Aho and Ullman [1]). In fact this has been done by Kurki-Suonio [27] who adds a symbol to the right of the righthand sides of the productions and Kuno [25] who adds a symbol to the left of the right-hand sides. Related ideas are in the definitions of parenthesis and bracketed grammars (see McNaughton [29] and Ginsburg and Harrison [10], respectively).
The special symbols can sometimes be considered as newly introduced nonterminal symbols which are left-hand sides of $\varepsilon$-productions.
For instance, this is done by Soisalon-Soininen [39] to convert the Kurki-Suonio idea to the cover formalism. Also Demers [7] does this to define generalized left corner parsing. A slightly restricted version of Brosgol's parse specifying translation grammar coincides with the simple SDTS of Definition 2.6. Demers uses Brosgol's definition. Following Demers, if we have a rule $A \to \alpha, h_\Sigma(\alpha_1 i \alpha_2)$ then $\alpha_1$ is called the left corner or leading part of the rule and $\alpha_2$ is its trailing part.

The following table lists a few names of parses which have been introduced before. Conform Demers [7] all production directed parses should be called *generalized left corner parses*. Notice that left part parses are defined as the opposite of left corner parse. Left part parses come close to being right parses.

| | |
|---|---|
| $i.A \to \alpha$, $h_\Sigma(i\alpha)$ | left parses |
| $i.A \to \alpha$, $h_\Sigma(\alpha i)$ | right parses |
| $i.A \to \alpha$, $h_\Sigma(\alpha_1 i \alpha_2)$ where $\alpha_1\alpha_2 = \alpha$ and $\lvert\alpha_1\rvert = 1$ | left corner parses (Rosenkrantz, Lewis [37]) |
| $i.A \to \alpha$, $h_\Sigma(\alpha_1 i \alpha_2)$ where $\alpha_1\alpha_2 = \alpha$ and $\alpha \in \Sigma^*$ or $\alpha_1 \in \Sigma^*N$ | extended left corner parses (Brosgol [6]) |
| $i.A \to \alpha$, $h_\Sigma(\alpha_1 i \alpha_2)$ where $\alpha_1\alpha_2 = \alpha$ and $\lvert\alpha_2\rvert = 1$ | left part parses (Nijholt [29]) |

Table II. Parses

The following table lists a few names and notations for covers according to Definition 2.5. We use $\bar{r}$ to denote right parses (or parse relations).

| parse relations | | notation | name |
|---|---|---|---|
| f | h | $G'[f/h]G$ | f-to-h cover |
| left | left | $G'[l/l]G$ | left cover |
| left | right | $G'[l/\bar{r}]G$ | left-to-right cover |
| right | left | $G'[\bar{r}/l]G$ | right-to-left cover |
| right | right | $G'[\bar{r}/\bar{r}]G$ | right cover |

Table III. Covers

It will be convenient to have the possibility to talk about positions in the right-hand sides of productions. Therefore we have the following notation.

NOTATION 2.1. Let $j.A \to X_1X_2\ldots X_n$ be a production of a CFG G. The positions in the right-hand side are numbered according to the following scheme:

$$j.A \to [1]X_1[2]X_2\ldots[n]X_n[n+1]$$

For a given production directed parse relation each production has a fixed position in which its number is inserted, conform Definition 2.6. We use $\Gamma_G(j)$ (or simply $\Gamma(j)$) to denote this position. Hence, $\Gamma_G : \Delta_G \to \mathbb{N}$.

We conclude this section with a few remarks on our definition of cover. Assume that $g : f_{G'} \to h_G$ is a cover homomorphism with $g = \langle\phi,\psi\rangle$. We have defined g in such a way that for any $i \in \Delta_{G'}$, $\psi(i) \in \Delta_G^*$. It is possible to introduce restrictions, for example,
(a) $\psi(i) \in \Delta_G \cup \{\varepsilon\}$, or (b) $\psi(i) \in \Delta_G$.

The definition of *complete cover* in Gray and Harrison [12] can be compared with our definition of cover if we have for $\phi$ the identity and $\psi$ is defined according to restriction (a). We do not include Gray and Harrison's notion of (not necessarily complete) *cover* in our formalism. Obviously it is possible to do so but since we already allow $\psi(i) \in \Delta_G^*$ we do not see useful applications.
It is also possible to include the notion of weak cover (Ukkonen [ ]). In that case one should allow homomorphisms to be defined on a subset of $f_G$,. For some other concepts of grammatical similarity the reader is referred to Hunt and Rosenkrantz [24].

## 3. THE LEFT PART TRANSFORMATION TO GREIBACH NORMAL FORM.

Now that we have presented the general framework, we come to the second aim of this paper: its application for a relevant transformation. We consider a transformation from *non-left-recursive grammars* to *Greibach normal form* (GNF) grammars. A few historical notes are in order. Greibach [13] introduced this normal form and she showed that each context-free language has a context-free grammar in this normal form. In Greibach [14] a more simple method is introduced to obtain the same result. Some of the structure preserving properties of the latter method have been investigated in Hotz [19,22]. Rosenkrantz [36] uses a matrix equation method to snow that each context-free language has a CFG in GNF. Both Greibach and Rosenkrantz start with an -except for a few minor conditions- arbitrary CFG and transform it into a weakly equivalent CFG in GNF.
Often such a transformation is done in two steps (for example, see Hopcroft and Ullman [18] and Aho and Ullman [1]). The first step is a transformation from an arbitrary CFG to a non-left-recursive grammar. The second step transforms the non-left-recursive grammar to a CFG in GNF. This second step (Aho and Ullman's Algorithm 2.14, attributed by them to M.Paull) will be referred to as the usual method. This usual method has been used in Wood [43] (see also Wood [44] and some of its structure preserving properties have been discussed in Kuno [25], Hotz and Claus [23], Benson [2] and for an adapted version, Nijholt [30].
There are some important subclasses of the non-left-recursive grammars for which more or less adapted versions of this usual method have been introduced. For example, the *LL(k) grammars* (Rosenkrantz and Stearns [38]) and the *strict deterministic grammars* (Geller, Harrison and Havel [9]).
The algorithm which will be considered in this section is a generalized version of a transformation which has been used in Nijholt [29,31].

Before we can introduce the algorithm we have to define a few less familiar concepts.

DEFINITION 3.1. Let $G = (N,\Sigma,P,S)$ be a CFG. Define a relation $CH \subseteq V \times N^*\Sigma$ as follows:
If $X_o \in N$ then $CH(X_o)$, the set of *chains* of $X_o$, is defined by

$$CH(X_o) = \{X_oX_1\ldots X_n \in N^*\Sigma \mid X_o \underset{L}{\Rightarrow} X_1\psi_1 \underset{L}{\Rightarrow} \ldots \underset{L}{\Rightarrow} X_n\psi_n, \psi_i \in V^*,\ 1 \leq i \leq n\}.$$

and for $c \in \Sigma$, $CH(c) = \{c\}$.

The left part transformation which we display below is an one-step transformation, in the sense that each production of the new grammar is obtained in one step from the productions of the original grammar. Another example of such a transformation is that of strict deterministic grammars to their GNF-version (Geller, Harrison and Havel [9]). Chains will be used for the construction of the right-hand sides of the productions of the new grammar.
Consider the following example grammar G with productions $S \to A|B$, $A \to a$ and $B \to a$. *Any* transformation of G into GNF yields a CFG with only production $S \to a$. Since a cover homomorphism is surjective it follows that no such homomorphism can be defined. However, it can be shown that any $\varepsilon$-free non-left-recursive CFG G can be given an equivalent $\varepsilon$-free non-left-recursive CFG G' without single productions (i.e., productions of the form $A \to B$, where $A,B \in N$). If we introduce a special production $S_o \to S\perp$, where S is the start symbol of G and $\perp$ is an endmarker, then this can be done in such a way that $G'[1/1]G$ (cf.Nijholt [30]).
In what follows we assume that, if necessary, first the single productions are eliminated. Hence, the input grammar will be a *very proper* (that is, no useless symbols, $\varepsilon$-free, no single productions) and non-left-recursive CFG.
The transformation is such that the new grammar left-to-x covers G, where, intuitively, x may 'run' from left to left part in the production directed parse relations.

DEFINITION 3.2. Let $G = (N,\Sigma,P,S)$ be a CFG. Define $[N] = \{[Ai\alpha] \mid i.\ A \to \alpha\beta \text{ is in } P, \text{ for some } \beta \in V^*\}$ and define a homomorphism $\xi : [N]^* \to [N]^*$ by letting $\xi([Ai\alpha])$ is
(i) $\varepsilon$ if i. $A \to \alpha$ is in P,
(ii) $[Ai\alpha]$ if i. $A \to \alpha\beta$ is in P, where $\beta \neq \varepsilon$.

DEFINITION 3.3. Let $G = (N,\Sigma,P,S)$ be a CFG. Define relation $LP \subseteq N^*\Sigma \times \Delta^*$ as follows:
Let $\pi = X_oX_1\ldots X_n \in N^+\Sigma$. $LP(\pi)$, the set of *left production chains* of $\pi$, is defined by

$$LP(\pi) = \{i_oi_1\ldots i_{n-1} \in \Delta^* \mid X_o \underset{L}{\overset{i_o}{\Rightarrow}} X_1\psi_1 \underset{L}{\overset{i_1}{\Rightarrow}} \ldots \underset{L}{\overset{i_{n-1}}{\Rightarrow}} X_n\psi_n,\ \psi_j \in V^*,\ 1 \leq j \leq n\}.$$

If $\pi \in \Sigma$ then $LP(\pi) = \{\varepsilon\}$.

In the algorithm we use Notation 2.1.

ALGORITHM 3.1. (*Left Part Transformation*)

*Input.* A very proper, non-left-recursive CFG $G = (N,\Sigma,P,S)$ such that for each production $j.\ A \to \alpha$ in $P$ we have that $\Gamma(j)$ satisfies $1 \le \Gamma(j) \le |\alpha|$.

*Output.* A weakly equivalent CFG $G' = (N',\Sigma,P',[S])$ in GNF.

*Method.* $P'$ is the set of all productions introduced below. $N'$ will contain $[S]$ and all symbols of $[N]$ which appear in the productions.

(i) For each pair $(\pi,\rho)$, $\pi = SX_1\ldots X_n \in CH(S)$ and $\rho = i_0 i_1 \ldots i_{n-1} \in LP(\pi)$, add $[S] \to X_n\xi([X_{n-1}i_{n-1}X_n]\ldots[Si_0X_1])$ to $P'$.

(ii) Let $i.\ A \to \alpha X_0\phi$ be in $P$, $\alpha \neq \varepsilon$; for each pair $(\pi,\rho)$, where

$\pi = X_0X_1\ldots X_n \in CH(X_0)$

and

$\rho = i_0i_1\ldots i_{n-1} \in LP(\pi)$

add

$[Ai\alpha] \to X_n\xi([X_{n-1}i_{n-1}X_n]\ldots[X_0i_0X_1][Ai\alpha X_0])$ to $P'$. □

THEOREM 3.1.

Let $G = (N,\Sigma,P,S)$ be a very proper, non-left recursive CFG. Assume, for each production $j.\ A \to \alpha$ in $P$, $1 \le \Gamma(j) \le |\alpha|$.

Algorithm 3.1 yields a CFG $G'$ in GNF such that $G'[1/x]G$, where $x$ denotes the parse relation defined by $\Gamma$.

*Proof.* (Sketch) Let $T = (N,\Sigma,\Delta,R,S)$ be the simple SDTS defined on $G = (N,\Sigma,P,S)$ which performs the translation $x$. Define $T' = (N',\Sigma,\Delta,R',[S])$ on $G' = (N',\Sigma,P',[S])$ by the rules:

(1) $[S] \to X_n\xi([X_{n-1}i_{n-1}X_n]\ldots[Si_0X_1])$,

$j_0j_1\ldots j_{n-1}\ \xi([X_{n-1}i_{n-1}X_n]\ldots[Si_0X_1])$

for each corresponding production introduced in step (i) of the algorithm.

The $j_k$'s are defined by, for $0 \le k \le n-1$, $j_k = i_k$ if $\Gamma(i_k) = 1$, and $j_k = \varepsilon$ otherwise.

(2) $[Ai\alpha] \to X_n\xi([X_{n-1}i_{n-1}X_n]\ldots[X_0i_0X_1][Ai\alpha X_0])$,

$jj_0j_1\ldots j_{n-1}\xi([X_{n-1}i_{n-1}X_n]\ldots[X_0i_0X_1][Ai\alpha X_0])$

for each corresponding production introduced in step (ii) of the algorithm.

The $j_k$'s and $j$ are defined by, for $0 \le k \le n-1$,

$j_k = i_k$ if $\Gamma(i_k) = 1$, and $j_k = \varepsilon$ otherwise, and

$j = i$ if $|\alpha X_0| = \Gamma(i)$, and $j = \varepsilon$ otherwise.

Cover homomorphism $\psi$ is defined by mapping each production of $P'$ on the string $j_0j_1\ldots j_{n-1}$ or $jj_0j_1\ldots j_{n-1}$ of its corresponding rule in $R'$, obtained in (1) or (2),

respectively. Clearly, T' is semantically unambiguous and therefore $\psi$ is a function. The main task is now to prove that $\tau(T') = \tau(T)$. Then, if $(w,\pi') \in l_G$, it follows immediately that $(w,\psi(\pi')) \in x_G$. Moreover, by the definitions of T' and $\psi$ it follows also that if $(w,\pi) \in x_G = \tau(T)$ then there exists $(w,\pi') \in l_G$, such that $(w,\psi(\pi') = (w,\pi)$. Thus we may conclude that G'[l/x]G. The proof that $\tau(T') = \tau(T)$ is omitted here. □

Because of the condition $1 \leq \Gamma(j) \leq |\alpha|$ the parse relations defined by such $\Gamma$ are the left parses, the left part parses and 'everything in between'. We can slightly weaken this condition by defining $\Gamma(j) \leq |\alpha|$ if $\alpha : 1 \in N$ and $\Gamma(j) \leq |\alpha| + 1$ if $\alpha : 1 \in \Sigma$.
However, the condition prevents that the theorem says anything about a left-to-right cover. We return to this problem in the following section.

We conclude this section with a result on (ε-free) *strict deterministic grammars* (Harrison and Havel [16]). Strict deterministic grammars are non-left-recursive. Hence the question arises whether our transformation preserves strict determinism. This is indeed the case. Since strict deterministic grammars are unambiguous it is sufficient to demand that the input grammar is ε-free and it does not have useless symbols.

COROLLARY 3.1. (Left part transformation for strict deterministic grammars.)
Let G be strict deterministic under partition $\pi$. Then G' is strict deterministic under a partition $\pi'$ which is defined as follows:
(i) $\Sigma \in \pi'$ and $\{[S]\} \in \pi'$
(ii) $[A i \alpha] \equiv [B j \beta] \pmod{\pi'}$ iff $A \equiv B \pmod{\pi}$ and $\alpha = \beta$

It follows that Theorem 3.1 can be used for strict deterministic grammars.
Note: It can be shown that partition $\pi'$ as defined above is the minimal strict partition of G'. Moreover, when the left part transformation is applied to a *real-time* strict deterministic grammar (Harrison and Havel [17]) the resulting grammar is also real-time strict deterministic.

## 4. OTHER COVERS

There remain some interesting questions. Firstly, in the preceeding section we obtained l/x-covers where (in an informal notation) $l \leq x \leq lp$. One can ask whether it is possible to replace lp by $\bar{r}$. This can not be done. That is, if we do not introduce restrictions on the (ε-free and non-left-recursive) grammars which we consider then lp is as far as we can go. In Ukkonen [41] an example of an ε-free and non-left-recursive grammar is given for which no left-to-right cover in GNF can be obtained.

Another question is whether 1 can be replaced by $\bar{r}$. This can be done. From the transitivity of the cover relation and from the following algorithm it follows that for any proper CFG G one can find a GNF-grammar G' such that $G'[\bar{r}/x]G$, with $1 \leq x \leq lp$.

ALGORITHM 4.1.

*Input.* A CFG G = (N,Σ,P,S) in GNF.

*Output.* A CFG G' = (N',Σ,P',S) in GNF such that $G'[\bar{r}/1]G$.

*Method.* In this algorithm each production in P' will be followed by its image under the cover homomorphism. If $A \in N$ then rhs(A) denotes its set of right hand sides. Initially set $P' = \{A \to a \ \langle i \rangle \mid i.A \to a \in P, a \in \Sigma\}$ and N' = N. The indexed symbols H which are introduced below are added to N'.

(i) For each production of the form $i.A \to a\alpha$ in P, $\alpha \neq \varepsilon$, the following is done.

Assume $\alpha = B\gamma$, $\gamma \in N^*$.

For any $j_k.B \to b_k\gamma_k$ in P, $1 \leq k \leq |rhs(B)|$

add

$$A \to aH_{ij_k}\gamma_k\gamma \ \langle\varepsilon\rangle$$

and

$$H_{ij_k} \to b_k \ \langle ij_k \rangle$$

to P'.

(ii) Remove all useless symbols. □

In the following corollary we collect the results of Algorithm 3.1, Algorithm 4.1 and the observation on the elimination of single productions.

COROLLARY 4.1.

Any proper non-left-recursive CFG G can be transformed to a CFG G' in GNF such that with $1 \leq x \leq lp$,

(i) $G'[1/x]G$, and

(ii) $G'[\bar{r}/x]G$.

In [33] a complete overview of cover results for $1/1$, $1/\bar{r}$, $\bar{r}/1$ and $\bar{r}/\bar{r}$-covers is given. For example, in (ii) we may drop the condition that G is non-left-recursive if we take $x = \bar{r}$. If we use non-right-recursive instead of non-left-recursive then we may take $x = \underline{r}$ in (i). A similar result (for unambiguous grammars) has been obtained by Ukkonen [40].

REFERENCES.


[1] AHO, A.V. and ULLMAN, J.D., *The Theory of Parsing, Translation and Compiling,* Vol.1 and 2, Prentice Hall, Englewood Cliffs, N.J. 1972 and 1973.

[2] BENSON, D.B. *Some preservation properties of normal form grammars,* SIAM J. Comput. 6 (1977), pp. 381-402.

[3] BERTSCH, E., *An observation on relative parsing time,* J.Assoc.Comput. Mach. 22 (1975), pp. 493-498.

[4] BOCHMAN, G.V., *Semantic equivalence of covering attribute grammars,* Publication #218, december 1975, Université de Montreal.

[5] BOCHMAN, G.V., *Semantic attributes for grammars with regular expressions,* Publication #195, Université de Montreal.

[6] BROSGOL, B.M., *Deterministic translation grammars,* Proc.Eight Princeton Conference on Information Sciences and Systems, 1974, pp. 300-306.

[7] DEMERS, A.J., *Generalized left corner parsing,* Conf. Record of the Fourth ACM Symposium on Principles of Programming Languages, 1977, pp. 170-182.

[8] FOSTER, J.M., *A syntax improving program,* Computer Journal 11 (1968), pp.31-34.

[9] GELLER, M.M., HARRISON, M.A. and HAVEL, I.M., *Normal forms of deterministic grammars,* Discrete Mathematics 16 (1976), pp.313-322.

[10] GINSBURG, S. and HARRISON, M.A., *Bracketed context-free languages,* J.Comput. System Sci. 1 (1967), pp. 1-23.

[11] GRAY, J.N. and HARRISON, M.A., *Single pass precedence analysis,* IEEE Conf.Record of the 10 th Annual Symposium on Switching and Automata Theory, 1969, pp. 106-117.

[12] GRAY, J.N. and HARRISON, M.A., *On the covering and reduction problems for context-free grammars,* J. Assoc.Comput.Mach. 19 (1972), pp. 385-395.

[13] GREIBACH, S.A., *A new normal-form theorem for context-free phrase structure grammars,* J. Assoc.Comput.Mach. 12 (1965), pp. 42-52.

[14] GREIBACH, S.A. *Erasable context-free languages,* Information and Control 29 (1975), pp. 301-326.

[15] GRIFFITHS, T.V., and PETRICK, S.R., *On the relative efficiencies of context-free grammar recognizers,* Comm. ACM 8 (1965), pp. 289-300.

[16] HARRISON, M.A., and HAVEL, I.M., *Strict deterministic grammars,* J.Comput. System Sci. 7 (1973), pp. 237-277.

[17] HARRISON, M.A., and HAVEL, I.M., *Real-time strict deterministic languages,* SIAM J. of Comput. 4 (1972), pp. 333-349.

[18] HOPCROFT, J.E., and ULLMAN, J.D. *Formal Languages and Their Relation to Automata,* Addison Wesley Publishing Co., Reading, Mass., 1969.

[19] HOTZ, G., *Normal-form transformations of context-free grammars,* to appear in Acta Kybernetica.

[20] HOTZ, G., *Eine Algebraisierung des Syntheseproblem von Schaltkreisen, I und II,* Elektr. Inform. und Kybernetik 1 (1965), pp. 185-231.

[21] HOTZ, G., *Eindeutigkeit und mehrdeutigkeit formaler Sprachen,* Elektr. Inform. und Kybernetik 2 (1966), pp. 235-246.

[22] HOTZ, G., *LL(k)-und LR(k) Invarianz von kontextfreien Grammatiken unter einer Transformation auf Greibachnormalform,* manuscript, 1978.

[23] HOTZ, G., and CLAUS, V., *Automaten-Theorie und Formale Sprachen III*, Bibliographisches Institut Mannheim, West Germany 1971.

[24] HUNT, H.B., and ROSENKRANTZ, D.J., *Complexity of grammatical similarity relations*, Proc. of the Conference on Theoretical Computer Science, Waterloo 1977, pp. 139-145.

[25] KNUTH, D.E., *Semantics of context-free languages*, Math. Systems Theory 2 (1968) pp. 127-145.

[26] KUNO, S., *The augmented predictive analyzer for context-free languages-Its relative efficiency*, Comm. ACM 9 (1966), pp. 810-823.

[27] KURKI-SUONIO, R., *On the top-to-bottom recognition and left recursion*, Comm. ACM 9 (1966), pp. 527-528.

[28] McNAUGHTON, R., *Parenthesis grammars*, J. Assoc.Comput.Mach. 14 (1967), pp.490-500.

[29] NIJHOLT, A., *On the parsing and covering of simple chain grammars*, in: Automata, Languages and Programming, G.Ausiello and C.Böhm (eds.), Lect.Notes in Comp.Sci.62 (Springer, Berlin, 1978), pp.330-344.

[30] NIJHOLT, A., *Grammar functors and covers: From non-left-recursive to Greibach normal form grammars*, to appear in BIT 19 (1979).

[31] NIJHOLT, A., *A left part theorem for grammatical trees*, Discrete Mathematics 25 (1979), pp. 51-64.

[32] NIJHOLT, A., *From left regular to Greibach normal form grammars*, unpublished manuscript.

[33] NIJHOLT, A., *A survey of normal form covers for context-free grammars*, IR-49, February 1979, Vrije Universiteit, Amsterdam.

[34] REYNOLDS, J.C., *Grammatical covering*, Argonne National Laboratory, T.M. No. 96, 1968.

[35] REYNOLDS, J.C. and HASKELL, R., *Grammatical coverings, unpublished manuscript*, Syracuse University, 1970.

[36] ROSENKRANTZ, D.J., *Matrix equations and normal forms for context-free grammars*, J.Assoc.Comput.Mach. 14 (1967), pp. 501-507.

[37] ROSENKRANTZ, D.J. and LEWIS, P.M. *Deterministic left-corner parsing*, IEEE Conf. Record of the 11th Annual Symposium on Switching and Automata Theory, 1970, pp. 139-152.

[38] ROSENKRANTZ, D.J. and STEARNS, R.E., *Properties of deterministic top-down grammars*, Information and Control 17 (1970), pp. 226-256.

[39] SOISALON-SOININEN, E., *On the covering problem for left-recursive grammars*, to appear in Theor.Comput.Science.

[40] UKKONEN, E., *Transformations to produce certain covering grammars*, in: Mathematical Foundations of Computer Science, J.Winkowski (ed.), Lect. Notes in Comp.Sci.64 (Springer, Berlin, 1978), pp. 516-525.

[41] UKKONEN, E., *Remarks on the nonexistence of some covering grammars*, to appear in Proc. 4th G.I.Conference on Theoretical Computer Science, 1979.

[42] WALTER, H.K.G. KEKLIKOGLOU, J. and KERN, W., *The behaviour of parsing time under grammar morphisms*, RAIRO-Theor.Comput.Science 12 (1978), pp.83-97.

[43] WOOD, D., *The normal form theorem - another proof*, Computer Journal 12 (1969), pp. 139-147.

[44] WOOD, D., *A generalized normal form theorem for context-free grammars*, Computer Journal 13 (1970), pp. 272-277.